Shot-based object retrieval from video with compressed Fisher vectors

SHOT-BASED OBJECT RETRIEVAL FROM VIDEO WITH COMPRESSED FISHER VECTORS

Luca Bertinetto, Attilio Fiandrotti, Enrico Magli

Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino
luca.bertinetto@polito.it, attilio.fiandrotti@polito.it, enrico.magli@polito.it

ABSTRACT

This paper addresses the problem of retrieving those shots from a database of video sequences that match a query image. Existing architectures are mainly based on Bag of Words model, which consists in matching the query image with a high-level representation of local features extracted from the video database. Such architectures lack however the capability to scale up to very large databases. Recently, Fisher Vectors showed promising results in large scale image retrieval problems, but it is still not clear how they can be best exploited in video-related applications. In our work, we use compressed Fisher Vectors to represent the video-shots and we show that inherent correlation between video-frames can be proficiently exploited. Experiments show that our proposal enables better performance for lower computational requirements than similar architectures.

Index Terms— object retrieval, object search, video search

1. INTRODUCTION

The widespread popularity of multimedia-enabled devices has fostered the blooming of large collections of digital items. As such collections grow larger, techniques for media searching, indexing and retrieval capable to scale up to very big databases are sought. For example, internet video providers such as YouTube are facing the challenge of how to efficiently answer users requests with appropriate video contents. To a smaller scale, domestic users may want to easily dig into libraries of videos that they collected over time. In the following, we refer to the application of retrieving those sequences from a database of videos that match a query image as *object retrieval from video*.

In their seminal *Video Google* paper [1], Sivic *et al.* proposed an object retrieval architecture based on the *Bag of Words* (BoW) model. Local features are extracted from a subset of the video-frames and represented as SIFT descriptors [2]. Then, using K-means, clusters of descriptors are created, forming a *vocabulary of visual words*. As we will detail in Section 2, images are represented as *visual word* frequency histograms over the vocabulary. Nevertheless, the vocabulary grows linearly with the number of frames in the database, thus making such architecture unsuited to handle large scale scenarios.

Conversely, Fisher Vectors (FVs) showed promising results in large scale image classification and retrieval problems [3, 4]. Images' local descriptors are represented with a Gaussian Mixture Model (GMM), where the Gaussians can be seen as the conceptual equivalent of the visual words in BoW. On the same performance basis, the number of Gaussians employed is orders of magnitude lower than the number of visual words required by the BoW model to achieve a reasonable performance.

In this paper, we introduce a novel architecture for object retrieval from video that improves over [1, 5] in the following aspects. *i)* The video is considered as a sequence of semantically coherent segments, the *shots*, rather than as a simple sequence of pictures. Within each shot, persistent descriptors are tracked so to represent the video through a few, yet highly distinctive, descriptors. *ii)* Shots are modeled via a FV-based representation, where each FV is further compressed by means of Principal Component Analysis (PCA). We compare with a reference architecture similar to [1, 6] and we show that the per-shot representation reduces offline learning time, while compressing the vectors reduces online memory requirements and query time without performance penalties.

The remainder of this document is structured as follows. Section 2 describes the BoW approach to video as first proposed in [1] and then refined in [5]. Section 3 describes our proposed architecture for object retrieval from video based on feature tracking and compressed FVs. Finally, in Section 4 we compare our architecture both with BoW and several possible FV configurations.

2. BACKGROUND

In this section, we overview the Video Google architecture described in [1, 5]. It can be summarized into an initial *offline* stage for the creation of a visual vocabulary, followed by an *online* stage for the processing of the queries.

2.1. Offline Stage

First, local features are located in a subset of the frames of the video database (the *keyframes*) and represented with SIFT descriptors [2]. Descriptors are clustered into K centroids using the K-means algorithm, whose output is the set $\mathcal{V} = \{v_1, \ldots, v_K\}$ of 128-dimensional elements, which we refer to as a *vocabulary of visual words*.

Second, each keyframe is represented with a K-bins histogram by mapping each descriptor extracted to its nearest neighbour in $\mathcal{V}$. TF-IDF weighting is then applied to the histograms to down-weight words that appear often in the vocabulary and to promote words that are highly characteristic of a specific keyframe. Eventually, each keyframe t_f, $f \in [1, F]$, is represented as a K-bins histogram and the resulting collection of F histograms is represented as the $K \times F$ matrix $\mathcal{T} = [t_1, \ldots, t_F]$.

2.2. Online Stage

The online stage is performed every time the user submits a query image. Descriptors are extracted and a histogram-based representation of the image is generated, mapping each descriptor to its nearest neighbour in $\mathcal{V}$. The histogram of the query image is then *TF-IDF* weighted. Let t_q be the $K \times 1$ vector representing the query image histogram: we compute the $F \times 1$ *score vector* $S^\top = [s_1, \ldots, s_F]$

$$S = \mathcal{T}^\top t_q, \tag{1}$$

where each value is the result of the inner product of t_q and t_f. The higher the value, the more likely it is that the f-th keyframe is similar to the query image, so sorting the score vector is equivalent to sorting the keyframes in order of relevance with the user query.

Finally, we would like to comment on the computational complexity of the BoW model. In [1, 5], the authors show that the number of visual words used is a fraction of the number of local descriptors. Therefore, the memory required to store $\mathcal{V}$ increases with the database size, together with the time spent during the online stage to map the new local descriptors to the visual words. Therefore, since BoW model is inherently limited in scalability, more compact and efficient retrieval architectures are sought.

3. PROPOSED ARCHITECTURE

In this section, we describe the proposed object retrieval architecture illustrated in Figure 1.

3.1. Shots representation

3.1.1. Shot Segmentation

We choose to represent each video as a collection of *shots*, where a shot is a sequence of frames that are coherent by content. The motivation behind this choice is twofold. First, it is highly likely that the user wishes to detect in which shot the query object appears, rather than in which particular frame. Second, grouping local descriptors at a shot-level allows to focus the representation on the non-redundant information, as it will be detailed in Section 3.1.3.

To detect the boundaries between consecutive shots, we simply compare the difference between the sum of the absolute difference in the color histograms of two consecutive frames with a threshold value. This first segmentation returns a subdivision of the video sequence in shots. Next, we apply the same segmentation process to each shot, but with a lower threshold. The result is a further subdivision of each shot: we consider as *keyframe* the median frame in each group of frames found by this second segmentation step.

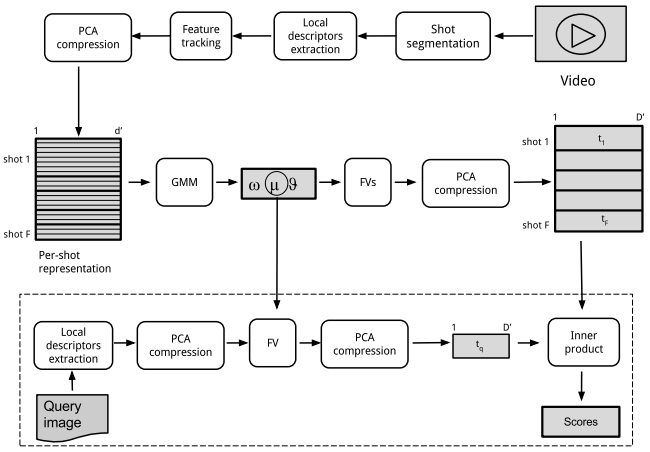

Fig. 1. Complete pipeline of our proposed architecture. Dotted line encloses the query stage described in Section 3.2.

3.1.2. Local Features Extraction and Representation

For each keyframe we compute Hessian-Affine regions [7] and we represent them via SIFT descriptors. Then, we convert them in *RootSIFT* [8] first by L_1-normalizing them and then computing the element-wise square root. Calculating the Euclidean distance between RootSIFT is in fact equivalent to calculating the Bhattacharyya distance with a Hellinger kernel on standard SIFT, which was shown to yield better results in several applications [8].

3.1.3. Feature Tracking

Due to the temporal correlation between consecutive keyframes, local descriptors tend to repeat across every shot, unnecessarily increasing the computational complexity of the FVs generation process. Aiming to represent each shot with the minimum amount of redundancy, we perform a per-shot tracking of local descriptors throughout the video. The objective is to obtain a representation of the development of

local descriptors across a shot, called *thread* in the following.

For each couple of consecutive keyframes k_{i-1} and k_i within a video-shot of length L, $i \in [2, L]$, for all the N_i local descriptors d^i_j, $j \in [1, N_i]$, we compute the first and second nearest neighbours and the respective euclidean distances in the local descriptors' space, respectively: d^i_{j1}, d^i_{j2}, Δ^i_{j1} and Δ^i_{j2}. If $r_j = \frac{\Delta^i_{j1}}{\Delta^i_{j2}} \leq \theta$, d^i_{j1} and d^i_j are considered as a *match*.

The motivation behind this operation is explained in [9], where the authors have studied the probability density function (PDF) of the ratio r_j on several thousands of images. They discovered that the PDF for correct matches significantly differs from the one for incorrect ones, and that, in general, incorrect matches tend to have a ratio near to 1.

Whenever a new match is obtained, d^i_{j1} is added to the same thread of d^i_j, otherwise d^i_{j1} is considered as a part of a new thread. Finally, each thread composed by more than one descriptor is represented as a *RootSIFT* that is the mean of all the local descriptors belonging to it. Doing so, the total number of local descriptors used to represent a shot is significantly reduced. This, as it will be illustrated in Section 4, permits an important reduction of the time needed to compute the FVs. As a very last step, the dimensionality of the threads is reduced from $d = 128$ to $d' = 64$ through PCA, as in [6] the authors show how such a reduction is not detrimental for the performance of an image retrieval system.

3.1.4. The Fisher Vector Representation of a Shot

Borrowing the notation from [3], let $X = \{x_1, \dots x_N\}$ be the set of N descriptors representing all the threads in the video database. The problem is to find the GMM of K_G multivariate Gaussians that best fits X. Solving this problem is equivalent to finding the set of d'-dimensional parameters $\lambda = \{w_1, \mu_1, \sigma_1, \dots, w_{K_G}, \mu_{K_G}, \sigma_{K_G}\}$ that maximizes the log-likelihood function

$$\mathcal{L}\{X|\lambda\} = \sum_{n=1}^{N} \log \sum_{k=1}^{K_G} w_k p_k(x_n|\lambda), \tag{2}$$

that is we want to find parameter vector λ that maximizes the likelihood that X was generated from a linear combinations of K_G multivariate Gaussian functions. This problem is iteratively solved by means of the Expectation Maximization algorithm.

Then, let $\mathcal{S} = \{s_1, \dots, s_T\}$ be the set of *RootSIFT* descriptors representing the threads of a shot: it can be described according to the GMM model with the gradient vector

$$G^{\mathcal{S}}_{\lambda} = \nabla_{\lambda}\, log\, p(X|\lambda). \tag{3}$$

Intuitively, it describes the direction in which the statistic model parameters should be modified to match the descriptors of the shot considered. The FV representation is obtained by concatenating all the gradient vectors relative to the K_G Gaussians. For the details of how this gradient can be efficiently computed, we refer the reader to [3] and [4].
In our experiments, we verified that considering only the gradients with respect to the Gaussians mean values μ yields a more compact description for a negligible performance loss. Therefore, in the following a FV representation of a shot is a vector composed of $D = K_G\, d'$ elements.

Since regular patterns (*e.g.* the bricks of a wall) tend to generate *bursts* of descriptors that can polarize the GMM against those areas of the image that are of interest to the user, we power-normalize each FV to reduce the impact of such descriptors as $f(z) = sgn(z)|z|^{0.5}$. Thereafter, each FV is also L_2-normalized, to guarantee that if one of the FV of the database is used as a query, the best retrieved shot is exactly the FV of the shot used as a query.

As a last step, we reduce the FVs dimensionality by means of a further stage of PCA, as [6] shows that PCA-reduced FVs exhibit smaller footprint with little performance penalty. Let $D = K_G\, d'$ be the length of a FV, the PCA compacts the corresponding vector to $D' \ll D$, where typical D' lengths generally lie between 128 and 1024. In some cases, PCA compression even improves the performance, as we show in Section 4.2.

3.2. Query Stage

During the query stage, a set $\mathcal{Q}$ of d'-dimensional RootSIFT descriptors is computed and a FV representation t_q of the query image is obtained with the same procedure described above. Gradient vectors $G^{\mathcal{Q}}_{\lambda}$ computed with respect to the mean values μ of the GMM previously computed are calculated and concatenated to create the FV representation. As a last step, the FV is power-normalized, L_2-normalized and PCA-reduced to the same size D' of the FVs of the database.

Recalling the same notation used in Section 2.2, the database of video sequences is represented by the matrix $\mathcal{T} = [t_1, \dots, t_F]$, where F indicates this time the number of shots in the database. In this context, $\mathcal{T}$ is a $D' \times F$ matrix, where each column is the FV representation of one of the F shots in the database. Let t_q, *i.e.* the FV representation of the query image, be a vector of length D': we obtain a scored list of matching shots by means of the inner product $S = \mathcal{T}^{\top} t_q$. Figure 2 shows an example query and a portion of the ranked list of retrieved shots.

The use of the inner product to compare two FVs is motivated by Perronnin *et al.* in [4], where they analytically illustrate the analogies between FVs and the TF-IDF weighted histograms of BoW model, proving the efficacy of the inner-product as a measure of similarity.

4. RETRIEVAL EXPERIMENTS

In this section, we compare a FV-based architecture with a reference based on a classic BoW model, described in Sec-

tion 2. Furthermore, we also consider several possible setups, showing that our proposal, which adopts feature tracking and compressed FV representation of the shots, achieves the best results.

4.1. Experimental Setup

The database of video sequences has been generated from the full-length movie "Inside Job", which is 109 minutes long. We detected approximately 150k frames, 1200 shots, 5000 keyframes and 6 millions of descriptors. By comparison, the authors of [1] experimented with a subset of 48 shots accounting for about 10k frames and 200k descriptors extracted from the movies "Run Lola run" and "Groundhog Day".
We experiment with 15 query images depicting objects, buildings and logos found in the movie, but taken from different points of view and under different levels of illumination and background cluttering. Most importantly, queries are completely unrelated with the database, as they have been picked from the Internet. To our knowledge, such a database has never been presented in a retrieval from video scenario, so we will make it available soon.
Conversely, in [1] the query images are taken from the very same movies used to build the video database. While our preliminary experiments showed that this latter approach enables better retrieval performance, we believe that our procedure is not only more challenging, but it also better reflects a possible real case scenario.

The considered architecture setups are evaluated using mean average precision (mAP) as retrieval performance metric. In addition to this, also three cost metrics are adopted.
Learning time is the time required to track the features and to represent the shots as compressed FVs. For the BoW model, instead, it is represented by the time required to create the vocabulary and to generate the histograms of visual words.
Query time is the time required to represent the query image in the appropriate form (histograms of words or FVs) and to return a ranked list of shots to the user. We do not account for the time required to compute SIFT, as it is a constant bias.
In-memory set represents the memory required to answer to a query. For FV-based setups, it accounts for the GMM statistical parameters vectors and the FV representation of the video shots. For BoW, it amounts to the visual vocabulary plus the histograms.

4.2. Experimental Results

Figure 3 compares the mAP achieved by a FV-based architecture against BoW as a function of the length of the representation of a single shot. Such length corresponds to the number of rows of matrix $\mathcal{T}$ described in the previous sections. For BoW, this value is equal to the number of visual words K, while for the proposed architecture it corresponds to $K_G\ d'$, as no PCA-reduction of the FV dimensionality is considered in this preliminary experiment. The figure clearly

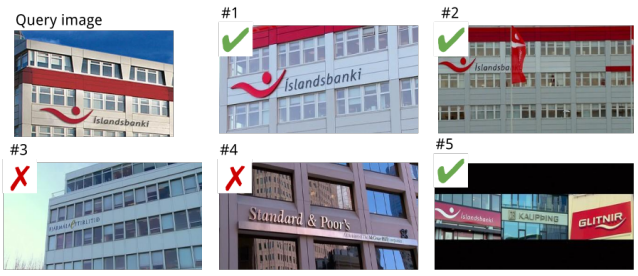

Fig. 2. An example of the first 5 shots retrieved for one of the query images. True positive matches are found in positions 1, 2 and 5. Notice how the false positive matches present very similar visual patterns to the query image.

shows that the FV-based architecture achieves the same mAP for a tenth of the database representation size required by the reference.

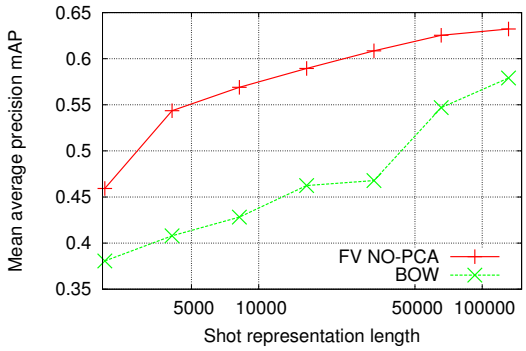

Fig. 3. Retrieval performance as a function of the length of each shot representation.

Then, in Figure 4 we compare different setups of a FV-based architecture. We report the mAP as a function of the number K_G of Gaussians for different degrees of FV compression. Clearly, the performance increases with the number of Gaussians. Less intuitively, the figure shows that PCA compression even increases the mAP at same values of K_G. *E.g.*, for $K_G \geq 512$, PCA-512 compression boosts the mAP by about 3%. This finding is coherent with the results of [6], and our understanding is that FVs bear a lot of internal correlation that is detrimental and that PCA removes.

Table 1 compares the computational requirements of the architectures described so far to achieve a mAP of about 0.6. BoW is affected by very large memory requirements and long learning and query times, which grow with the number of visual words K. Moreover, since the Nearest Neighbour search is responsible for most of the time spent during the retrieval stage, K also drives the query time.

FV-based architectures exhibit much lower computational

	K or K_G	Shot representation length	mAP	Learning Time	Query Time	In-memory set
BoW	131072	131072	0.582	1600 min	4.6 s	1.2 GB
FV NO-PCA	1024	65536	0.625	51 min	0.079 s	319 MB
FV NO-PCA	512	32768	0.616	23.5 min	0.034 s	159 MB
FV PCA-512	512	512	0.634	24 min	**0.029 s**	**2.8 MB**
FV F.Tracking PCA-512	512	512	0.635	**12 min**	**0.029 s**	**2.8 MB**

Table 1. Computational and memory requirements of the evaluated architectures for comparable values of mAP. Our proposed architecture (last row) boasts lowest memory requirements thanks to FV compression and lowest learning time thanks to feature tracking. Tests were performed on 16-cores Intel Xeon @2.90GHz.

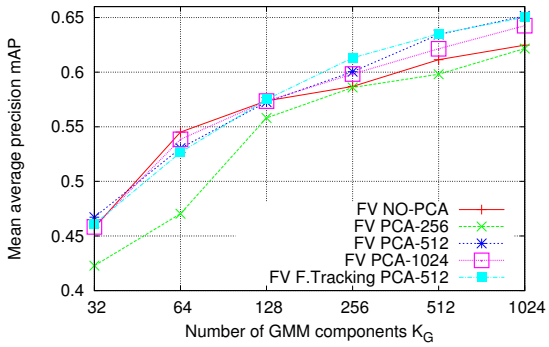

Fig. 4. Retrieval performance of different setups as a function of the number of Gaussians K_G.

time and smaller memory footprint at similar retrieval performance. For K_G=1024 and without PCA compression (*FV NO-PCA*), all the costs considered are significantly reduced. Further reducing K_G from 1024 to 512 compacts the shot representation length, which is equal to $K_G d'$, thus obtaining a GMM that is less complex to fit and a smaller matrix $\mathcal{T}$ that is faster to query. The second to last row shows the beneficial effects of PCA compression. When FV length is reduced from $D = K_G d'$ = 32768 to D'=512, *i.e.* by a factor of 64, the in-memory set decreases by the same amount and the learning time increases of just about 30 seconds due to the PCA compression. Finally, the last row presents the costs of our proposed architecture. Together with Figure 4, it shows that feature tracking is beneficial for the learning time and not detrimental for performance.

5. CONCLUSIONS

In this paper we proposed a novel architecture for object retrieval from video based on a compressed Fisher Vector representation of video-shots. We analysed the problem as initially introduced by [1], introducing and adapting the state of the art in image retrieval to a video scenario. On top of that, we demonstrate that exploiting the natural correlation between consecutive video-keyframes, it is possible to significantly reduce the computational time with no loss in terms of performance.

Acknowledgements

This work was supported by TELECOM Italia, research contract 7010067123.

REFERENCES

[1] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 1470–1477.

[2] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 2007.

[4] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, "Large-scale image retrieval with compressed fisher vectors," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 3384–3391.

[5] J. Sivic and A. Zisserman, "Efficient visual search for objects in videos," *Proceedings of the IEEE*, vol. 96, no. 4, pp. 548–566, 2008.

[6] H. Jégou, F. Perronnin, M. Douze, C. Schmid, et al., "Aggregating local image descriptors into compact codes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 9, pp. 1704–1716, 2012.

[7] K. Mikolajczyk and C. Schmid, "An affine invariant interest point detector," in *Computer VisionECCV 2002*, pp. 128–142. Springer, 2002.

[8] R. Arandjelovic and A. Zisserman, "Three things everyone should know to improve object retrieval," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2911–2918.

[9] S. Lepsoy G. Francini, M. Balestri, "Cdvs: Improved image comparison by weighted matching," *ISO/IEC JTC1/SC29/WG11 MPEG2011/M25795*, 2012.